International Journal of Computer Science Engineering
and Information Techonology Research (IJCSEITR)
ISSN (P): 2249-6831; ISSN (E): 2249-7943
Vol. 8, Issue 2, Jun 2018, 61-68

NATURE INSPIRED ALGORITHMS FOR TEST SUITE OPTIMIZATION FOR REGRESSION TESTING: AN INVESTIGATION

CHETAN J. SHINGADIYA[1] & NITESH M. SUREJA[2]

[1]*Research Scholar, Rai University, Ahmedabad, Gujarat, India*

[2]*Director, Om Engineering College, Junagadh, Gujarat, India*

ABSTRACT

Software Testing is one of the expensive activities in software development. But at the same time, it will assure the quality of the software product. Software testing will take more cost and maximum time which actually increases the cost of the product as well as wastage of time. We have investigated various approaches proposed to addresses the problems in test suite optimization using nature-inspired algorithms. Such problems can be solved using nature-inspired algorithms. This approach is based on Genetic Algorithm, Particle Swarm Optimization, Memetic Algorithm is investigated. The results obtained by different researchers are collected and comparative analysis is prescribed and listed.

KEYWORDS: Genetic Algorithm, Particle Swarm Optimization, Memetic Algorithm, Test Suite Minimization & Test Suite Optimization

Received: Apr 07, 2018**; Accepted:** Apr 28, 2018**; Published:** May 10, 2018**; Paper Id.:** IJCSEITRJUN20187

Original Article

INTRODUCTION

The increased visibility of software as a system element and the associated "costs" associated with a software failure are motivating forces for well-planned, in-depth testing. The software development organization spends a majority of the project effort on testing. The human-rated software can cost three to five times as much as all other software engineering steps together or even life if it fails. It is necessary to have a technique to test the entire system thoroughly and in less time. Therefore, Test Suite Optimization plays an important role in optimizing the test cases required to test the entire system. The main objective of the test-site optimization is to improve the quality of the product, reduce the time and effort involved in the testing phase, and reduce the cost of the software project. Software testing is a process of running software with the intention of finding errors. It is an expensive activity and takes a lot of time and effort. But it still remains the primary mechanism by which errors are detected and it provides assurance of the quality and reliability of the software.

By reducing the test suites, we can reduce the time and effort significantly. A good quality of the test suites lead to an early error detection and thus improves the regression tests. Test Suite Optimization techniques are used to prioritize and select test cases in a test suite. Some of the test cases in the original test-suite may be redundant with respect to the test criteria. To reduce the effort and cost of testing, the optimization objectives of the test suites should be selected to select only a minimal subset of the original test suite that still meets all testing requirements. [1].

Nature inspired computing is a technique that is inspired by nature's processes. These computer techniques led to the development of algorithms called Nature-Inspired Algorithms (NIA). These algorithms are subject to

computer intelligence. The purpose of developing such algorithms is to optimize technical problems.

As the world moves toward industrialization, technical issues become more complex and difficult to optimize. This is due to increasing dimensions, variables, temporal complexity, spatial complexity, etc. In order to cope with such a situation, nature-inspired algorithms have been developed. NIA is mainly categorized into evolutionary algorithms and swarm intelligence-based algorithms [9].

This paper is organized as follows the description and applicability of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Memetic Algorithms (MA), to optimization problems respectively. It provides the comparative investigation of algorithms applied to Test Suite Optimization also talk about results and at the end conclusions is described.

GENETIC ALGORITHM

A genetic algorithm is one of the most admired search techniques in which different applications solve the various problems of the real world. Some of the applications are available in the areas of optimization, search, and decision-making [2]. Genetic algorithms are currently the most prominent and commonly used computer models of evolution in artificial life systems. These decentralized models provide a foundation for understanding many other systems and phenomena in the world. Research on GA in one's life provides vivid examples that use the genetic algorithm to study how learning and evolution interact and to model ecosystems, the immune system, cognitive systems, and social systems [X].

Genetic Algorithms consist of four main parts, namely selection, crossover, mutation and updating. Each module includes the process to find the best solution among the available solutions [II] [XII]. The simple steps of the processes involved in the genetic algorithms are shown using given figure 1.

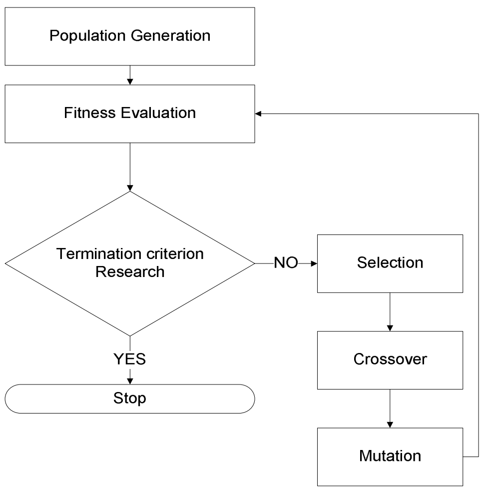

Figure 1: Flowchart of Genetic Algorithm [II]

In population phase, the possible symbols in the solution sequences are used to generate the population. Thus to generate the different combination of solutions the random process is used with the symbols. After each randomly generate population is tested by the fitness function. The fitness function is a kind heuristic which measures the generated population, whether the population sequences are providing solutions or not. Additionally, if that provides finding fitness of individual solution, the solution needs to optimize more. So, the termination criterion is evaluated first. In the genetic algorithm, two key termination conditions are frequently used [II].

- The maximum number of evaluation cycles is completed
- Or, the optimal solution is obtained

The pseudo code for GA is given below.

```
Initialize population

Evaluate population

While (! stop Condition) do {

        Select the best fit individuals for reproduction;

        Breed new individuals through crossover and mutation
        operations;

        Evaluate the individual fitness of new individuals;

        replace least fit population with new individuals;
```

Pseudo code of GA [VII] [XII]

PARTICLE SWARM OPTIMIZATION

A Particle Swarm Optimizer is a population-based stochastic optimization algorithm. It simulates the social behavior of bird flocks. Kennedy and Eberhart proposed it in 1995 [XIII] [XV]. PSO is similar to Genetic Algorithm in the sense that both approaches are population-based and each individual has a fitness function. Furthermore, the adjustments of individuals in PSO are relatively similar to the arithmetic crossover operator used in Genetic algorithms. However, PSO is influenced by simulation of social behavior rather than survival of the fittest.

In PSO, each individual benefits from its history, which is not there in Genetic algorithms. PSO has been widely used in a wide range of optimization problems as it is very easy to implement. It is also applied to some continuous nonlinear and discrete optimization problems by different authors. Every candidate solution is represented as an in PSO [III]. Particle swarm optimization is a global optimization algorithm for dealing with problems in which a best solution can be represented as a point or surface in a n-dimensional space. Hypothesis are plotted in this space and seeded with an initial velocity, as well as the communication channel between the particles [XV]. Particles then move through the solution space and are evaluated according to some fitness function after each timestamp. Over time particles are accelerated towards those particles within their grouping which have better fitness values. Each in the swarm is represented by the following characteristics:

- The current position of the particle
- The current velocity of the particle

The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a “flying” through the fitness landscape finding the maximum or minimum of the objective function. Initially, the PSO algorithm chooses candidate solutions randomly within the search space. The search space is composed of all the possible solutions. It should be noted that the PSO algorithm has

no knowledge of the underlying objective function used, and thus has no way of knowing if any of the candidate solutions are near to or far away from a local or global maximum. The PSO algorithm simply uses the objective function to evaluate its candidate solutions, and operates upon the resultant fitness values. Each maintains its position, composed of the candidate solution and its evaluated fitness and velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness, and the candidate solution that achieved this fitness, referred to as the individual best PSO algorithm maintains the best fitness value achieved among alls in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution [III].

The simple flowchart of PSO is given below.

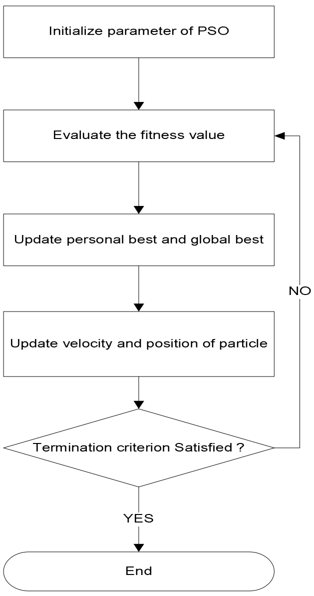

Figure 2: Flowchart of Particle Swarm Optimization [VII]

The simple pseudo -code for PSO is given below.

```
Initialize the population Do {
        For i=1 to Population Size
        {
                Calculate fitness value
                IF the fitness value is better than the best one (pBest) in history
                then set current value as the new pBest
        }
        Choose the particle with best fitness value of all the particle as gBest
        For i=1 to Population Size
        {
                Calculate new velocity
                Update particle position in accordance
                }
```

Pseudo code of PSO [VII]

MEMETIC ALGORITHM

Memetic Algorithm (MA), inspired by the Darwinian principles of natural evolution and Dawkins' notion of a meme, the term "Memetic Algorithm" (MA) was first introduced by Moscato in his 1989 technical report in which he considered MA to be closely related to a form of the Population-based Hybrid Genetic Algorithm (GA) coupled with an individualized learning process that can perform local refinements [XI]. The metaphorical parallels, on the one hand, with Darwinian evolution and, on the other hand, with memes and domain-specific (local search) heuristics are captured in memetic algorithms, creating a methodology that balances well between generality and problem specificity. The method has proved to be practically successful in a large number of problem areas and in particular for the approximate solution of optimization problems. The adjective "memetic" is derived from the term "meme", which is described by R. Dawkins as analogous to the gene in the context of cultural developments. The term "meme" is defined as "the basic unit of cultural transmission or imitation" [IV]. Memetic Algorithm (MA), The population is initialized at random or using a heuristic. Then, each individual makes a local search to improve its fitness. To form a new population for the next generation, higher quality individuals are selected. The selection phase is identical in form that used in the classical genetic algorithm selection phase.

Once two parents have been selected, their chromosomes are combined and the classical operators of crossover are applied to generate new individuals. The latter is enhanced using a local search technique. The role of local search in Memetic algorithms is to locate the local optimum more effective [IV].

Memetic Algorithm (MA) hybridizes global and local search, Such that the individuals of a population in a global search algorithm have the opportunity for local improvement in terms of local search. The use of MAs for test suite optimization was originally proposed in this paper. For procedural code, and has since then been applied in different domains. Although MAs have been already used in the past for a unit test generation, their applications have been mainly focused on numerical data types and covering specific testing targets (e.g., a branch) with a single test case or test suite.[V]

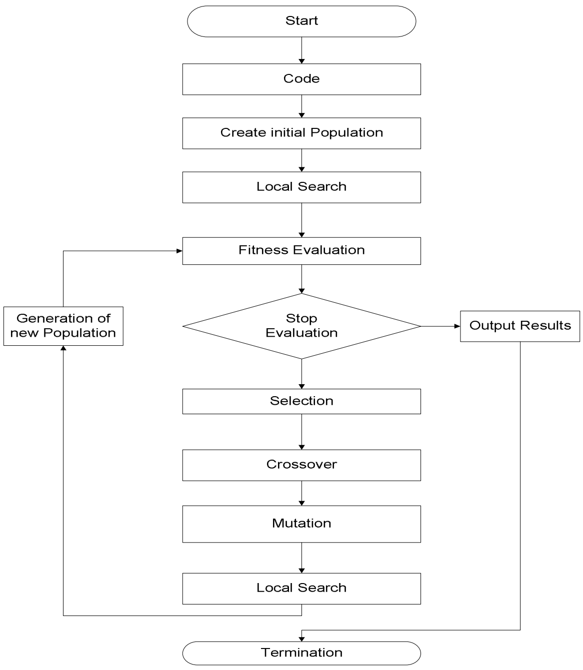

Figure 3: Flowchart of Memetic Algorithm [7]

Pseudo code for Memetic Algorithm is given below.

```
Initialize populations;
While (convergence criterion not reached)
{
        Evaluate the fitness function of every individual;
        Select Parents;
        Recombine to produce offspiring;
        Mutate offspring;
        Improve offspring via local search;
        Replace individual with new version;
        }
```

Pseudo Code of (MA) [IX]

COMPARATIVE INVESTIGATION

As per our investigation for Genetic Algorithm (GA), we found that researcher has shown the requirement specification for a ticket booking application is considered the analysis of performance. The results of the minimized test suite of transaction segment are presented in table 1.

Table 1: Result Analysis on Online Ticket Booking for Test Suite (GA) [VIII]

Transaction Segment	Original Test Suite	Minimized Test Suite
New User Registration	14	14
Ticket availability checking	67	57
Book a Ticket	50	38
Cancel a Ticket	20	16
Ticket Status Checking	15	12

Table 2 shows that the calculation time for application modules and analysis result of time taken for each transaction segment before and after minimization of the test suite.

Table 2: Result Analysis on Online Ticket Booking For Execution Time (GA) [VIII]

Transaction Segment	Execution Time for Original Test Suite (Min)	Execution Time for Minimized Test Suite (Min)
New User Registration	252	251
Ticket availability checking	807	683
Book a Ticket	994	763
Cancel a Ticket	197	158
Ticket Status Checking	208	168

As per our investigation for Memetic Algorithm(MA), we found that the researcher has taken two benchmarks for their research first is SF100 and second is Carfast, then they applied both benchmarks in Evosuite for 2 min per class with 10 iterations to accommodate randomness. Table 3 summarize result: on Carfast benchmark the use of local search cover on average 44,853 more branches then pure global search. On SF100 the increases are smaller; 693 additional branches

were covered by Memetic Algorithm.

Table 3: Comparison of GA with Best MA for both SF100 and Car Fast Case Studies

Case Study	Classes	Total	GA	MA
Car fast	1.392	1,18,1,234	513,669	558,521
SF100	11,088	238,760	93,600	94,240

The Average number of Covered Branches is Reported and The Difference between Two Configuration. [V]

As per our investigation for particle swarm optimization, we found that researcher has given study about finding out the maximum number among the three numbers to generate and optimize test suite. For that he has converted given program into CFG (Control, Flow Graph) and based CFG he has estimated the complexity of the program. After that, they had to design the test suite for given program using branch coverage and white box testing.

The table collected the test suite for the given program which are results of contains thirty numbers of test cases. After execution of test cases, there are some test cases which are repeated and some of them are unnecessary. So to avoid this problem PSO technique is used. The test cases may be passed through the PSO to optimize. [VI]

Table 4: Particle Swarm Optimization (PSO) [VI]

Iteration Number	No of Test Cases	Status
1	25	Mixed
2	20	Mixed
3	16	Optimized

RESULTS

Results obtained by various researchers in their research work for test suite optimization using nature-inspired algorithms are collected. These results are depicted in tabular fashion in tables from table 1 to 4. Results in each of above tables represent the value found from all algorithms applied to test suite optimization and good value for particular algorithms in certain criteria. We can analyze all algorithms based on the results summarized in tables.

While implementing GA its provide a good result when we have limited collection data but in case of the complex problem, it shows performance degradation also increases the complexity of a problem. GA leads to lack of stability and overall implementation is complicated for complex optimization as compared to other but give us good processing time. MA provides good result as compared to GA because it's overcome the issues of GA also it is focused on both local and global search. Memetic Algorithm (MA) has been used in the past for local search operators, branch coverage of classes may be significantly improved the result. In the end, Particle Swarm Optimization (PSO) may work in local and global minima problem.

CONCLUSIONS

This paper presents an investigation carried out for the performance analysis of some nature-inspired algorithms like Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Memetic Algorithm. Research papers based on algorithms are reviewed. All papers based on test suite optimization. Each paper examined considering various aspects of time, cost, complexity, efforts, and quality of output. The investigation clearly says that every algorithm has certain capabilities in them if they are explored with good criteria. We conclude that every algorithm has its own advantages and

disadvantages, at the same time they all have their importance in research work. Here we also thank you all the authors who provided us some very good research information in the form of their research paper to carry out these investigations.

REFERENCES

1. *Sara Amjad, Kuldeep Jaiswal. (2016). Survey of Optimization Techniques for Test Suite Optimization in Regression Testing. International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297:2007 Certified Organization) Vol. 4, Issue 2, pp. 1981-1983*
2. *Subham Kothari, Anand Rajavat. (2017). Minimizing The Size of Test Suite Using Genetic Algorithm for Object Oriented Program. IEEE. ICT in Business Industry & Government (ICTBIG), International Conference onTransactions on Software Engineering.pp-16-21*
3. *Nitesh Sureja, VedVyas Dwivedi. (2012). Nature Inspired Algorithms Based on Travelling Salesman Problems: An Investigation. © Inventi Journals (P)Ltd, Inventi Impact: Artificial Intelligence Vol.2012, Issue 3, pp. 126-129*
4. *T. Hashni, M. Divyavani. (2013). An study of effective approaches in nature inspired techniques for solving optimization problems. International Journal of Advance Research in Computer Science Vol. 4, No 2, pp. 143-145*
5. *Gordan Fraser, Andreau, Phil McMinn. (2014). A Memetic Algorithm for whole test suite generation. The Journal of system software, pp. 45-56.*
6. *Rasmita Gouda, SibaPrasadaTripathy. (2013). Application of Heuristic Technique PSO in Optimization of Structural Testing International Journal of Computing Technique and Application Vol. 1, No 2, pp. 35-39*
7. *Ijad Madisch, Soren Hofmayer, Horst Ficknscher. https://www.researchgate.net/*
8. *Srividhya Jeyaprakash, Dr. K. Algarsamy. (2015). A Distinctive Genetic Approach for Test Suite Optimization. The 2015 International Conference on Soft Computing and Software Engineering, pp. 427-434*
9. *Paul Agrwal, Shiksha Mehta. (2014). Nature-Inspired Algorithms: State-of Arts. Problems and Prospects, International Journal of Computer Application, Vol. 100, No 14, pp. 14-21*
10. *Sivanandan, S. N, S N Deepa.(2008). Introduction to Genetic Algorithms. Springer Publication, Newyork.*
11. *Saravanan, N., and R. Rathnasamy. "A New Tool using Simulation and Optimization of Solar Adsorption Cooling Technique."*
12. *Pablo Mascato, Carlos Cotta, Alexander Mendes (2004). MemeticAlgorithm"researchgate.net/publication, pp. 53-86*
13. *Pravin Ranjan Srivastava, Tai-hoon Kim. (2009). Application of Genetic Algorithm in Software Testing. International Journal of Software Engineering and its Application, Vol 3, No 4, pp. 87-95*
14. *Majhi, Bidya Prakash, and Shatendra Sahu. "A Rewiew on Application of Bio-Geography based Algorithm and Other Optimization Techniques."*
15. *James Kennedy and Russel Ebehart.(1995). Swarm Intelligence. IEEE, pp. 1942-1948*
16. *Yuhi Shi and Russel Ebehart.(1998). A Modified Swarm Optimizer. IEEE, pp. 69-73*
17. *Hemlata S. Urade and Prof. Rahila Patel (2011). Study and Analysis of Swarm Optimization: A Review. International of Computer Applications (@IJCA),, pp. 1-5*